WEST Search History

```
DATE:  Saturday, August 30, 2003
```

Set Name side by side	**Query**	**Hit Count**	**Set Name** result set
DB=USPT,PGPB; THES=ASSIGNEE; PLUR=YES; OP=OR			
L21	5729718[uref]	21	L21
L20	5854941[uref]	9	L20
L19	L18 and l17	1	L19
L18	transduc$ and servo	13349	L18
L17	(5729718 5854941).pn.	2	L17
L16	L15 or l14	20	L16
L15	L10 and l4 and l3	11	L15
L14	L13 or l6	20	L14
L13	L12 and l4 and l3	19	L13
L12	L8 and (hard near3 (disk or disc or drive) or (disc or disk) near4 drive)	119	L12
DB=JPAB,EPAB,DWPI,TDBD; THES=ASSIGNEE; PLUR=YES; OP=OR			
L11	L8 and (hard near3 (disk or disc or drive) or (disc or disk) near4 drive)	1	L11
DB=USPT,PGPB,JPAB,EPAB,DWPI,TDBD; THES=ASSIGNEE; PLUR=YES; OP=OR			
L10	L9 and (hard near3 (disk or disc or drive) or (disc or disk) near4 drive)	29	L10
L9	((serch or track or seek) near3 distance or skew) and cylinder and (command or access$2) near3 queue and head	29	L9
L8	((serch or track or seek) near3 distance or skew) and cylinder and queue and head	152	L8
DB=JPAB,EPAB,DWPI,TDBD; THES=ASSIGNEE; PLUR=YES; OP=OR			
L7	((serch or track or seek) near3 distance or skew) and cylinder and pending near5 (access$2 or operation or command) and head	0	L7
DB=USPT,PGPB; THES=ASSIGNEE; PLUR=YES; OP=OR			
L6	L5 and l2	5	L6
L5	L4 and l3	679	L5
L4	(radial near5 distance or seek) with cylinder	2133	L4
L3	head near5 (move$ or seek$3)	101669	L3
L2	L1 and (hard near3 (disk or disc or drive) or (disc or disk) near4 drive)	52	L2
L1	((serch or track or seek) near3 distance or skew) and cylinder and pending near5 (access$2 or operation or command) and head	53	L1

```
END OF SEARCH HISTORY
```

End of Result Set

☐ Generate Collection

L11: Entry 1 of 1 File: TDBD Sep 1, 1998

TDB-ACC-NO: NNRD413132

DISCLOSURE TITLE: DASD Command Reordering Algorithm That Accounts for Tangential Position

PUBLICATION-DATA:
Research Disclosure, September 1998, UK

VOLUME NUMBER: 41
ISSUE NUMBER: 413
PUBLICATION-DATE: September 1, 1998 (19980901)

CROSS REFERENCE: 0374-4353-41-413-0

DISCLOSURE TEXT:

This document contains drawings, formulas, and/or symbols that will not appear on line. Request hardcopy from ITIRC for complete article. Disclosed is a Direct Access Storage Device (DASD), or disk drive, command reordering algorithm that reduces average service times. The algorithm does this by minimizing access time of all three spatial access dimensions. It executes faster and can make better choices by utilizing an initial presort that minimizes the access time of only two dimensions. The algorithm also lends itself to self optimization to account for access times, for a given movement, that can vary over time.

At some time prior to when a host system starts requesting a disk drive do reads and writes, parameters used to determine command beginning and ending spatial positions and access times between those positions are calculated and saved. For example, head switch and cylinder switch formatted skew times are needed to determine the tangential position of requested blocks. This information could be calculated and saved at manufacturing, format, and/or power up times in order to reduce overhead during read/write command processing. Seek and settle times for all potential accesses need to be determined also. This information can be hard-coded (fixed) or be made changeable so that the data can be updated at a later time with more accurate information. Thus if for any reason servo performance changes from drive to drive, or from time to time for the same drive, the drives can self-optimize the reordering algorithm dynamically. As the commands are received by the drive, the beginning and ending spatial positions are determined and saved for later reordering use. Those algorithms use the values saved at initialization. The algorithms themselves are a function of the format architecture of a particular disk drive. Typical current reordering designs do the reordering as the commands arrive and are enqueued, or placed in the queue.

Those reordering algorithms do not, nor do they need to, take into account tangential position. To save processor bandwidth, thus command overhead, those reordering algorithms can be removed since another reordering algorithm will eventually execute that does take all three positions into account. But executing a relatively quick reordering algorithm when commands are enqueued that groups commands together that are radially close together can make a subsequent re-reordering algorithm do two things... 1. execute faster 2. make better choices.

For example, when re-reordering taking tangential position into account, it can be advantageous to limit the number of potential candidates to select from to some number less than the total number that may be in the command queue at that time. This allows that process to execute in less time. This also increases the likelihood that the best candidate in the entire command queue is located in the shortened queue that is actually scanned. Note: It is advantageous to use an algorithm here that increases the the likelihood of grouping commands into regions that have minimized range between the innermost and outermost radial positions. Such an algorithm may or may not create the minimum radial access time possible.

The only part of the algorithm relating to the execution of the command that gets altered by this invention is the point in time when a subsequent command is dequeued, or popped, from the queue to be executed. The rest may remain the same. At the point in time a new command is dequeued for execution, instead of executing the command predetermined to be next at the time the commands were enqueued, a search is added to find another command in the queue. That command's access time, including radial, vertical and tangential portions, is the smallest of the commands in the queue at that time.

A scan, starting with the command due next to be executed, or 'head' of queue, or a command near the head, is made looking for the minimum access time. The access times for commands are calculated with respect to a fixed command. Where the fixed command is located in the queue slot scheduled to be executed immediately prior to where the scan is started. For example the fixed command may be the command that just finished executing. While scanning, a pointer is retained that points to the command that has the smallest access time of the commands scanned up to a given point in time. The comparison made within the scanning loop to find that command is made in the time domain, rather than a logical domain, in order to account for all 3 spatial dimensions.

After the scan has completed, the command currently being pointed to by the 'smallest access time' pointer is pulled out of it's queue position and inserted into the queue position immediately following the command used to calculate access time with. At this point, the tangential position reordering algorithm is complete and command processing carries on as before. The only difference being that a command with shorter total access time may have been scheduled to execute after the 'fixed' command than what was previously scheduled. Below is a pseudo code example of the scan. SEE ORIGINAL.